

Отчет по практической работе №3

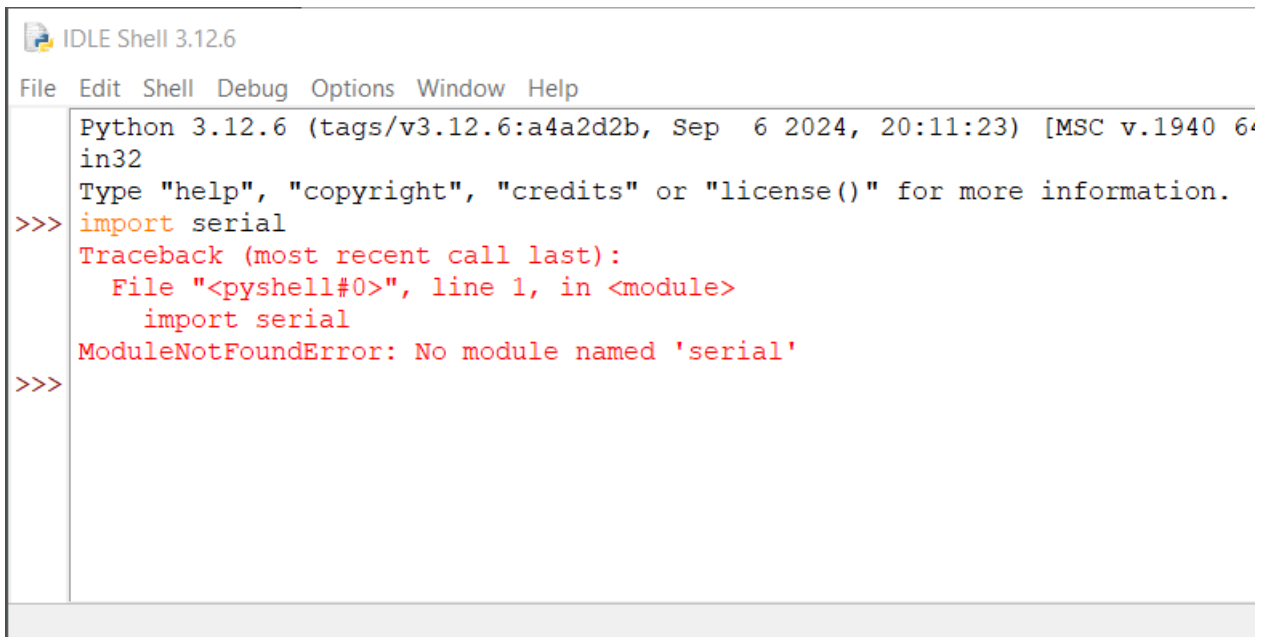
Реализация программы работы с последовательным портом средствами Python

Цель работы:

Реализовать программу для работы с последовательным портом, изучить основы работы с Anaconda и Jupyter Notebook

Ход работы:

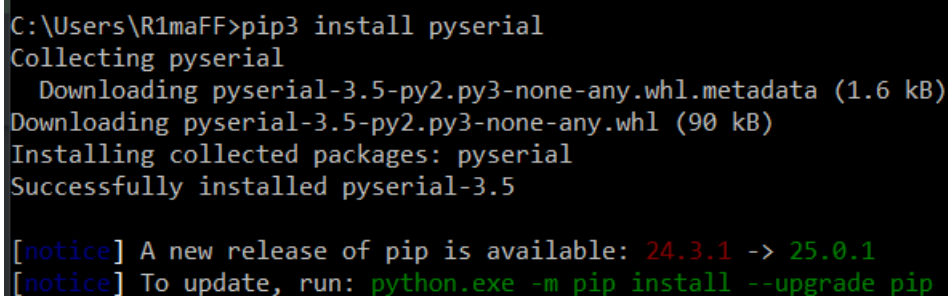
Запускаем Python IDLE и проверяем наличие библиотеки pyserial.



```
Python 3.12.6 (tags/v3.12.6:a4a2d2b, Sep 6 2024, 20:11:23) [MSC v.1940 64-bit x86_64]
Type "help", "copyright", "credits" or "license()" for more information.
>>> import serial
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import serial
ModuleNotFoundError: No module named 'serial'
>>>
```

Рис 3.1

Библиотека не установлена, поэтому загружаем ее через cmd.

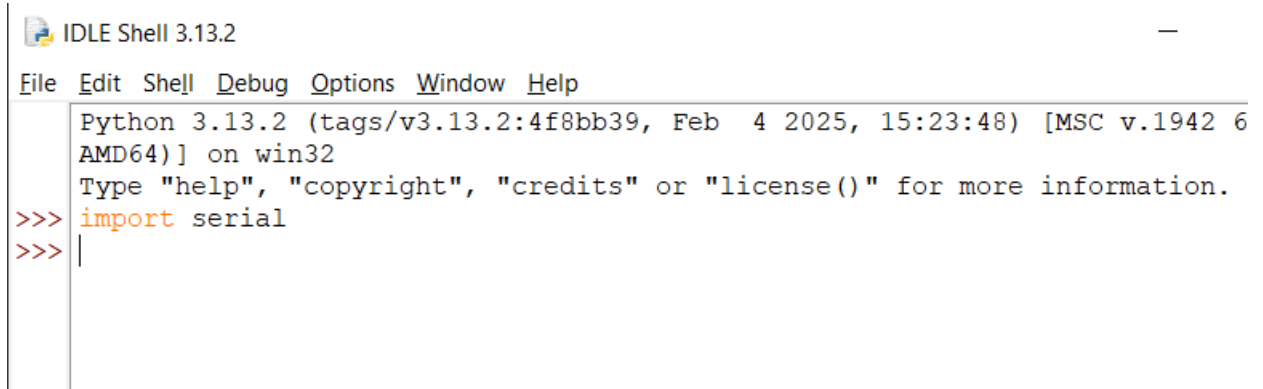


```
C:\Users\R1maFF>pip3 install pyserial
Collecting pyserial
  Downloading pyserial-3.5-py2.py3-none-any.whl.metadata (1.6 kB)
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
Successfully installed pyserial-3.5

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Рис 3.2

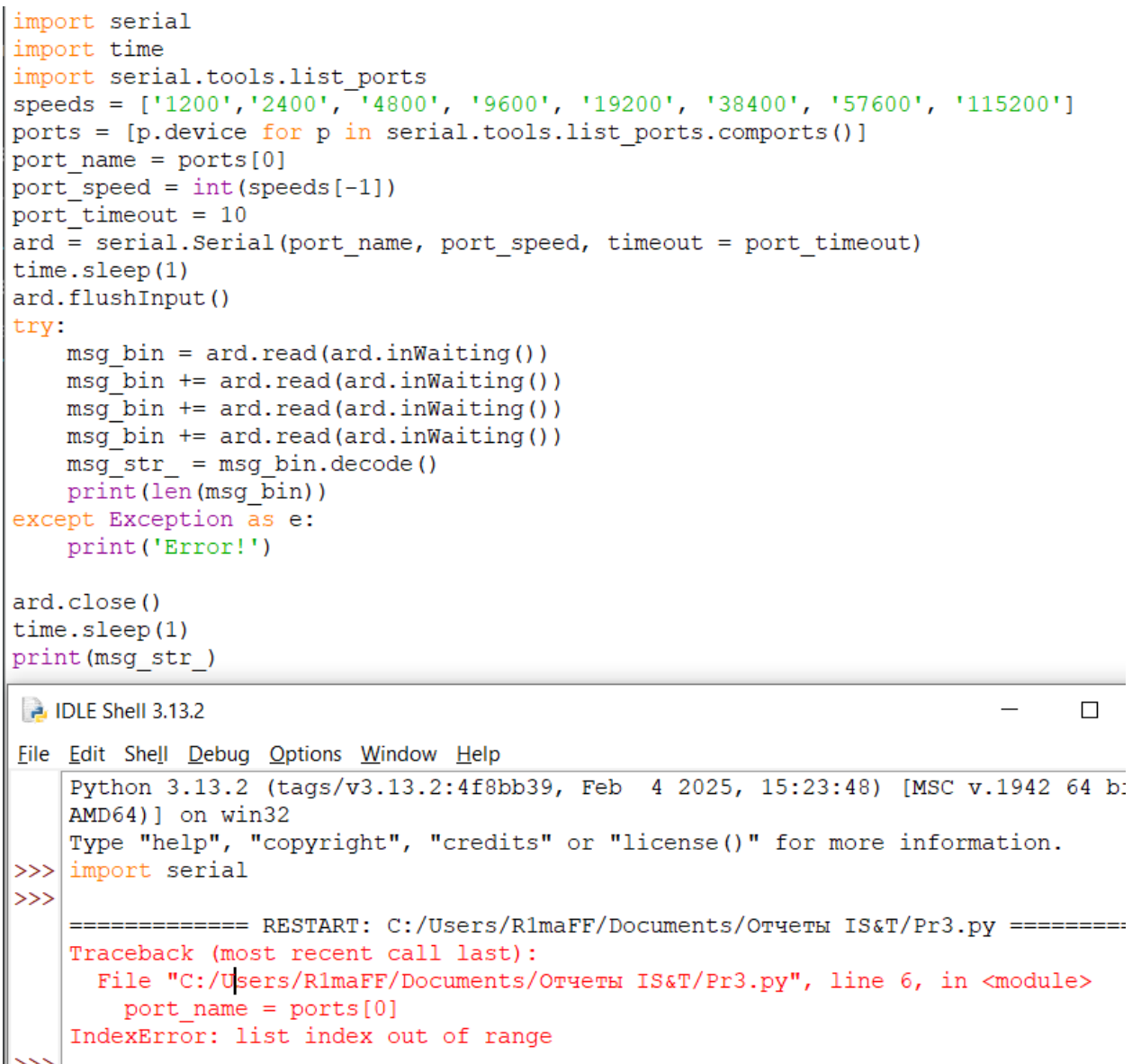
Теперь при импорте библиотеки никаких ошибок не выходит.



```
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 6
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import serial
>>> |
```

Рис 3.3

Создаем в IDLE новый файл и вставляем код программы. При запуске программа выдает ошибку, что нет доступных последовательных портов.



```
import serial
import time
import serial.tools.list_ports
speeds = ['1200', '2400', '4800', '9600', '19200', '38400', '57600', '115200']
ports = [p.device for p in serial.tools.list_ports.comports()]
port_name = ports[0]
port_speed = int(speeds[-1])
port_timeout = 10
ard = serial.Serial(port_name, port_speed, timeout = port_timeout)
time.sleep(1)
ard.flushInput()
try:
    msg_bin = ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_str_ = msg_bin.decode()
    print(len(msg_bin))
except Exception as e:
    print('Error!')

ard.close()
time.sleep(1)
print(msg_str_)
```

```
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 b:
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import serial
>>>
===== RESTART: C:/Users/RlmaFF/Documents/Отчеты IS&T/Pr3.py =====
Traceback (most recent call last):
  File "C:/Users/RlmaFF/Documents/Отчеты IS&T/Pr3.py", line 6, in <module>
    port_name = ports[0]
IndexError: list index out of range
>>>
```

Рис 3.4

Для решения этой ошибки необходимо эмулировать последовательные порты. Я использовал com0com. С помощью этой программы создаем 2 порта, к которым далее будем подключаться.

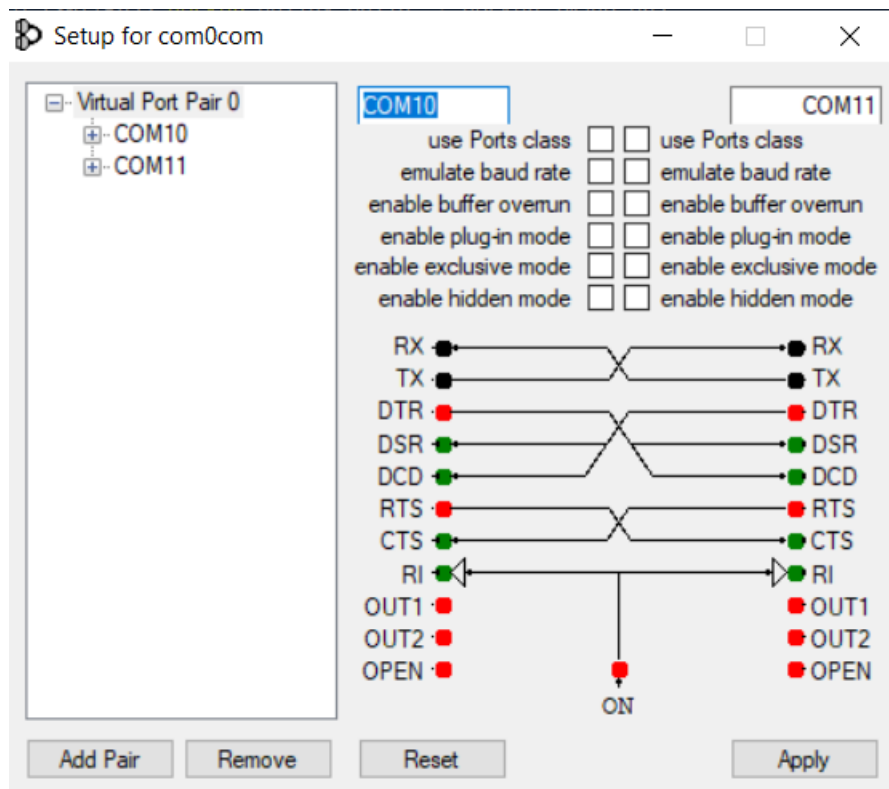
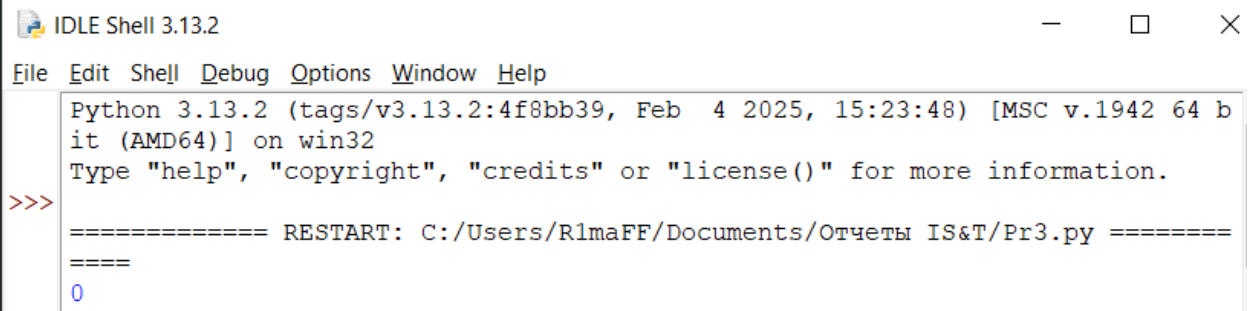


Рис 3.5

Теперь если установить в программе вместо `ports[0]` значение одного из COM портов, а подключиться к другому, то выполнение пройдет без ошибок, но мы не сможем передавать данные в программу.

```
import serial
import time
import serial.tools.list_ports
speeds = ['1200', '2400', '4800', '9600', '19200', '38400', '57600', '115200']
ports = [p.device for p in serial.tools.list_ports.comports()]
port_name = 'COM3'
port_speed = int(speeds[-1])
port_timeout = 10
ard = serial.Serial(port_name, port_speed, timeout = port_timeout)
time.sleep(1)
ard.flushInput()
try:
    msg_bin = ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_bin += ard.read(ard.inWaiting())
    msg_str = msg_bin.decode()
    print(len(msg_bin))
except Exception as e:
    print('Error!')

ard.close()
time.sleep(1)
print(msg_str_)
```



```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/RlmaFF/Documents/Отчеты IS&T/Pr3.py =====
====
0
```

Рис 3.6

Для передачи данных в программу необходимо немного изменить ее код. Нужно добавить бесконечный цикл, в котором программа будет ожидать получения данных от второго порта. Завершение программы реализовано через сочетания клавиш Ctrl + C.



```
Pr3.py - C:\Users\R1maFF\Documents\Отчеты IS&T\Pr3\Pr3.py (3.13.2)
File Edit Format Run Options Window Help

# Импорт библиотек
import serial
import time
import serial.tools.list_ports

# Доступные скорости передачи данных
speeds = ['1200', '2400', '4800', '9600', '19200', '38400', '57600', '115200']

# Получение списка доступных портов
ports = [p.device for p in serial.tools.list_ports.comports()]

# Настройка имени порта и скорости передачи
port_name = 'COM10'
port_speed = int(speeds[3])
port_timeout = 10
try:
    # Открытие последовательного порта
    ard = serial.Serial(port_name, port_speed, timeout = port_timeout)
    ard.flushInput()
    time.sleep(1)
    print("Ожидание данных...")

    # Бесконечный цикл для чтения данных
    while True:
        if ard.inWaiting() > 0:
            msg_bin = ard.read(ard.inWaiting())
            msg_str = msg_bin.decode('utf-8')
            print({msg_str})
        else:
            time.sleep(0.1)
except Exception as e:
    print('Error!')

# Завершение программы с помощью сочетания клавиш Ctrl + C
except KeyboardInterrupt:
    ard.close()
    print("Программа завершена.")
```

Рис 3.7

Для передачи данных в программу я использовал Terminal 1.9b. Настраиваем программу на COM10, а терминал на COM11.

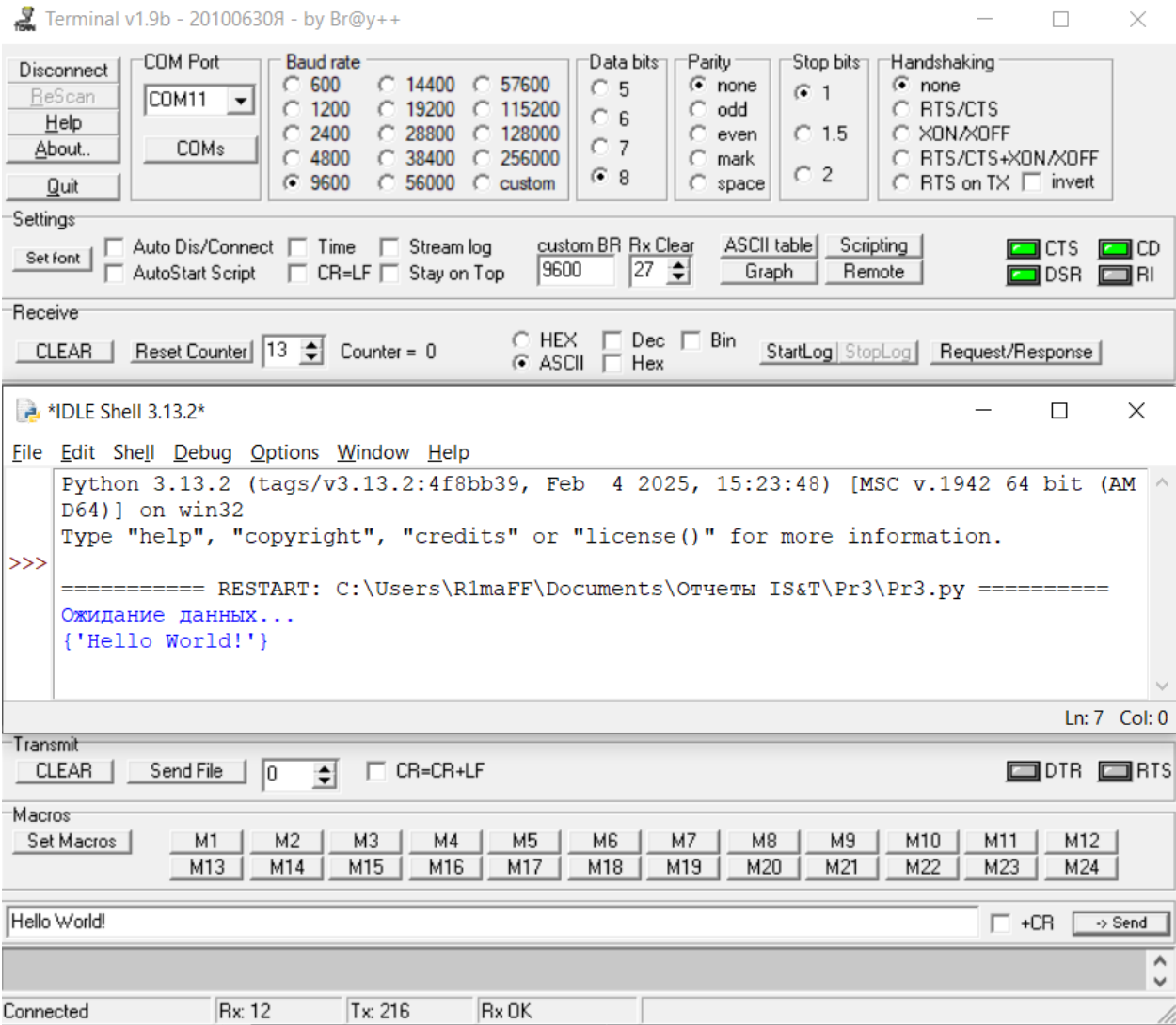


Рис 3.8

Далее необходимо создать `docker_image` данной программы.

```
PS C:\Users\R1maFF\Documents\Отчеты IS&T\Pr3> docker buildx build -t docker_image_pr3 .
[+] Building 25.6s (11/11) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 187B                                0.0s
=> [internal] load metadata for docker.io/library/python:3.13-slim 2.7s
=> [auth] library/python:pull token for registry-1.docker.io       0.0s
=> [internal] load .dockerignore                                   0.0s
=> => transferring context: 2B                                       0.0s
=> [1/5] FROM docker.io/library/python:3.13-slim@sha256:f3614d98f38b0525d670f287b0474385952e28eb43016655dd003d0 18.2s
=> => resolve docker.io/library/python:3.13-slim@sha256:f3614d98f38b0525d670f287b0474385952e28eb43016655dd003d0e 0.0s
=> => sha256:b7f4d10c88643f0a5e68db62e3670060d5d8442343cf515b850e21dec58b6982 12.58MB / 12.58MB 1.5s
=> => sha256:b33772645d099be9d3bab4f00333f537a43f78f64c98cbcb9dda1ae5d8857137 250B / 250B 1.6s
=> => sha256:082b22c6387c101db186aedb7e557ba697c3d8a5cd4f5ac166559653a3faa128 3.51MB / 3.51MB 2.3s
=> => sha256:7cf63256a31a4cc44f6defe8e1af95363aee5fa75f30a248d95cae684f87c53c 28.22MB / 28.22MB 16.8s
=> => extracting sha256:7cf63256a31a4cc44f6defe8e1af95363aee5fa75f30a248d95cae684f87c53c 0.7s
=> => extracting sha256:082b22c6387c101db186aedb7e557ba697c3d8a5cd4f5ac166559653a3faa128 0.1s
=> => extracting sha256:b7f4d10c88643f0a5e68db62e3670060d5d8442343cf515b850e21dec58b6982 0.4s
=> => extracting sha256:b33772645d099be9d3bab4f00333f537a43f78f64c98cbcb9dda1ae5d8857137 0.0s
=> [internal] load build context                                   0.1s
=> => transferring context: 1.36kB                                     0.0s
=> [2/5] WORKDIR /app                                             0.2s
=> [3/5] COPY requirements.txt .                                  0.1s
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt       3.1s
=> [5/5] COPY . .                                                 0.1s
=> exporting to image                                             1.0s
=> => exporting layers                                              0.6s
=> => exporting manifest sha256:f0efa4c8cd35ac1901f66fffd47c10bf230c00b598e4d78bc64da282832c4897d 0.0s
=> => exporting config sha256:2b6dc864724f3201e6232d81049c451b993c96150b7ad9b46ea6ec90ae816d30 0.0s
=> => exporting attestation manifest sha256:720d56e3d25fb057ff150c46645e7f78faa8e97e4f144485ba42d7d1177bc285 0.0s
=> => exporting manifest list sha256:a3019f64abd4024bdc7de15d1fd990e38957a7a70c2e5c059d3fb01cae867316 0.0s
=> => naming to docker.io/library/docker_image_pr3:latest         0.0s
=> => unpacking to docker.io/library/docker_image_pr3:latest      0.2s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/q8vetliv7cjnn4v4t3jhw4x4h
```

Рис 3.9

Теперь необходимо запустить `docker container`. С этим возникают сложности. Так как контейнер Docker запускается на Linux, то он не видит COM портов. При работе с Linux необходимо обращаться к `ttyS`, а не к `COM`, из-за этого контейнер не запускается и выдает ошибку.



Рис 3.10

Далее я решил попробовать поработать с портами через WSL (Windows Subsystem for Linux). Проверил наличие последовательных портов с помощью `ls /dev/tty*`, но ни один из выданных вариантов не соответствует `COM10`. Скорее всего из-за того, что WSL не эмулирует COM порты напрямую, как это делает Linux.

Я попробовал использовать `com2tcp` для проброски порта.

```
C:\WINDOWS\system32>com2tcp --baud 9600 \\.\COM10 0.0.0.0 9000
OpenC0C("\\.\COM10", baud=9600, data=8, parity=no, stop=1) - OK
```

Рис 3.11

Com2tcp запустился без каких-либо ошибок, теперь осталось подключиться к это порту через WSL с помощью socat, но у меня это так и не получилось. Socat выдает Connection Refused.

```
root@DESKTOP-70GSH4V:~# socat TCP:172.28.176.1:9000 PTY,link=/dev/ttyS10,b9600,raw,echo=0
2025/02/25 18:29:05 socat[1894] E connect(5, AF=2 172.28.176.1:9000, 16): Connection refused
```

Рис 3.12

Основы Anaconda и Jupyter Notebook

Устанавливаем Anaconda Navigator, запускаем Jupyter Notebook и создаем новую папку.

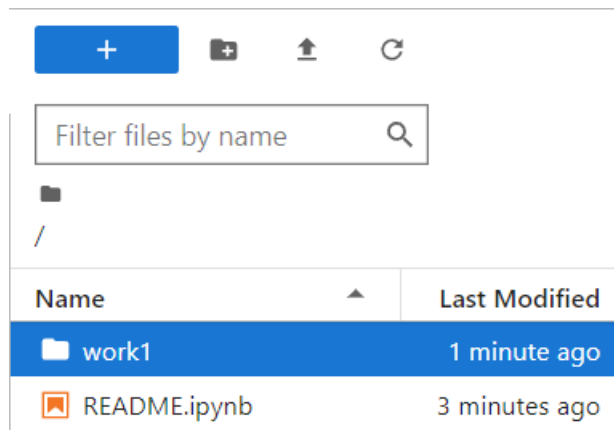


Рис 3.13

Создаем новый Python файл.



Рис 3.14

Открываем этот файл, набираем код и проверяем работу программы.

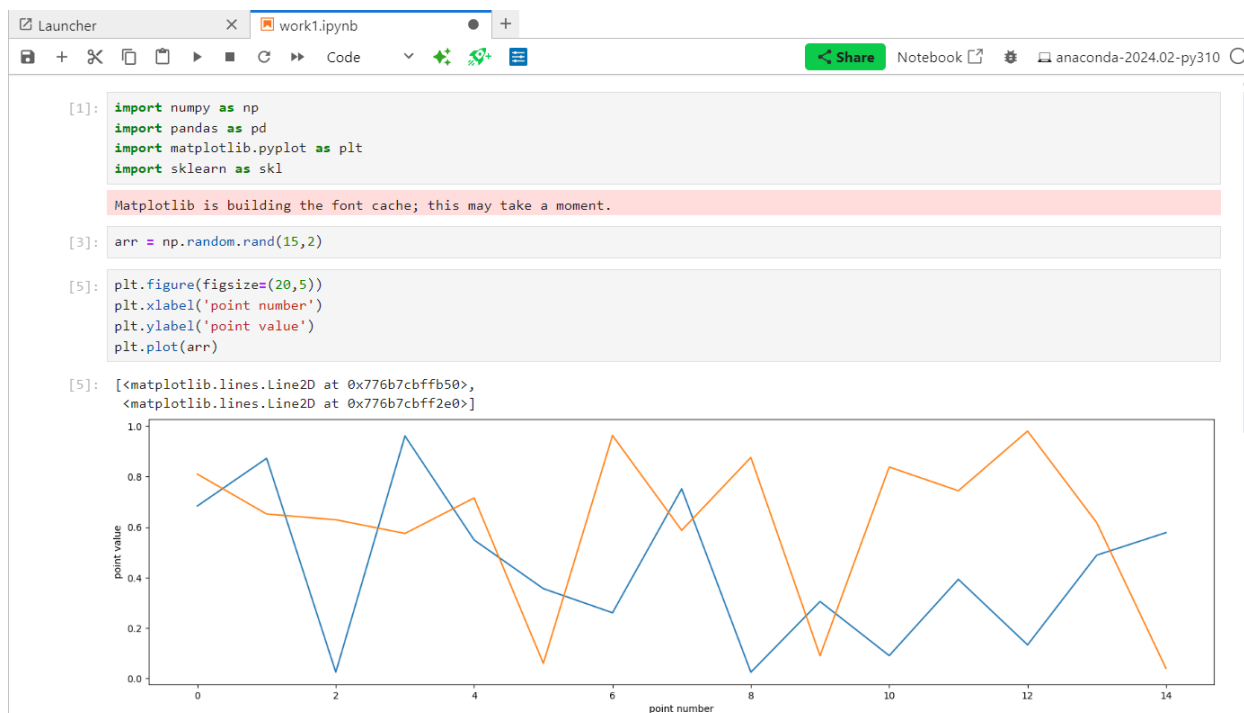


Рис 3.15

Вывод: В результате практической была реализована программа для работы с последовательными портами, изучены основы работы с Anaconda и Jupyter Notebook. Была предпринята попытка починки программы для работы с последовательными портами путем создания виртуальных портов, изменения кода программа и передачи данных с помощью терминала. Получилось наладить работу программы локально, но не получилось наладить ее работу через docker container. Я думаю, что нужно было изначально работать через WSL и создавать виртуальные порты через socat, а не через com0com.